Unit –3: Constructor and Destructor

Constructor:

- Constructor is a special function that is automatically called when an object of a class is created.
- Constructor has the same name as the class.
- Constructor does not have a return type. They don’t return anything. Constructor is automatically called by the compiler and it is normally used to initialize values.
- Constructor is useful for initialize values of objects.

Characteristics of constructor:

- Constructor has the same name as class name.
- They should be declared in the public section of the class declaration
- They are invoked automatically when the objects are created.
- They don’t have return type.
- They can’t return value.
- They can have default arguments.
- Construction can’t be virtual.
- We cannot refer to their address.
- They make implicit call to the memory management operator(new and delete) when memory allocation is required.
- Constructors can be overloaded.

Define and initialize constructor

- Constructor is a special function having the same name as that of its class, that is automatically called when an object of a class is created.
- It is used to initialize some values to the data members of an object.

Syntax:

```
Class_name (list-of-parameters)
{
        //constructor definition
}
```

Example:

```
class point
{
        private:
                int x;
                int y;
        public:
        point( )                 // constructor definition
        {
                x=0;            // Initialize data member of class
                y=0;
        }
};
point:: point()
{
                x=0;            // Initialize data member of class
                y=0;
}
```

Types of Constructors in C++

Default constructor

- A default constructor is a constructor that has no parameters.
- If it has parameters, all the parameters have default values.
- It is also called a zero argument constructor.

Example:

```
class point
{
        private:
                int x;
                int y;
        public:
        point( )
        {
                x=0;
                y=0;
        }
        void putpoint( )
        {
                cout<<x<<y<<endl;
        }
};
```

```
void main( )
{
        point p1;
        p1.putpoint( );
 }
```

Output:0 0

Constructor with arguments (parameterized constructor)

- The constructor which can take the arguments that is called as parameterized constructors.
- It follows all properties of the constructor.
- It takes parameters to initialize the data.
- The parameterized constructor is called by two ways:

Implicit calls : point p2(5,7);

Explicit calls : point p1 = point(5,7);

Example:

```
class point
{
        private:
                int x;
                int y;
        public:
                point( )
                {
                        x=0;
                        y=0;
                }
                point(int x1,int y1)
                {
                        x=x1;
                        y=y1;
                }
```

```
void putpoint( )
{
        cout<<x<<y<<endl;
}
};
void main( )
{
        point p1;
        point p2(5,7);
        p1.putpoint( );
        p2.putpoint( );
}
```

Output:

0,0

5,7

Copy Constructor

- The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.
- It is used to initialize one object from another of the same type.
- It Copy an object to pass it as an argument to a function.
- It Copy an object to return it from a function.

```
class point
{
        private:
        int x;
        int y;
        public:
        point(int x1,int y1)
        {
                x=x1;
                y=y1;
        }
        point(point& p)
        {
                x=p.x;
                y=p.y;
        }
```

```
void putpoint( )
{
        cout<<x<<y<<endl;
}
};
void main( )
{
        point p2(5,7);
        p2.putpoint( );
        point p3(p2);
        p3.putpoint( );
        point p4=p3;
        p4.putpoint( );
        point p5;
        p5=p4;
        p5.putpoint( );
}
```

Overloading constructor (multiple constructors)

- Overloaded constructors have the same name (exact name of the class) and different by number and type of arguments.
- A constructor is called depending upon the number and type of arguments passed.
- While creating the object, arguments must be passed to let compiler know, which constructor needs to be called.
- It is also known as multiple constructor.

Array of object using constructors

- The array of objects represents storing multiple objects in a single name.
- Each element in the array is an instance of the class.
- Each one's member variables can have a unique value.
- This makes it possible to manage and handle numerous objects by storing them in a single data structure and giving them similar properties and behaviours.

```
#include<iostream>
class Employee
{
        int id;
        char name[30];
        public:
        void getdata()
        {
        cout << "Enter Id : ";
        cin >> id;
        cout << "Enter Name : ";
        cin >> name;
        }
void putdata()
{
cout << id << " "<<name <<endl;
}
};
```

```
void main()
{
Employee emp[4];
int i; S
for(i = 0; i < 4; i++)
{
        emp[i].getdata();
}
cout << "Employee Data - " << endl;
for(i = 0; i < 4; i++)
{
        emp[i].putdata();
}
}
```

Constructor with default arguments

- A default arguments are those arguments which have a value assigned at the time of the function declaration.
- Default arguments of the constructor are those arguments which have a value assigned at the time of constructor declaration.
- If the values are not provided when calling the constructor the constructor uses the default arguments automatically.

```
class point
{
        private:
        int x;
        int y;
        public:
        point( )
        {
                x=0;
                y=0;
        }
        point(int x1,int y1=0)
        {
                x=x1;
                y=y1;
        }
void putpoint( )
{
cout<<” ( “<<x<<” , ”<<y<<” ) ”<<endl;
} };
```

```
void main( )
{
        point p1;
        point p2(5);
        point p3(5,7);
        p1.putpoint( );
        p2.putpoint( );
        p3.putpoint( );
}
```

Output:

(0,0)

(5,0)

(5,7)

Destructor:

- Destructors in C++ are members functions in a class that delete an object.
- Destructors have the same name as their class and their name is preceded by a tilde(~).
- Destructors don’t take any argument and don’t return anything.
- They are called when the class object goes end of scope such as when the function ends or the program ends.
- Destructor is useful for releasing memory of objects.

Characteristics of destructor:

- Destructor has the same name as that of the class prefixed by the tilde character '~'.
- They should be declared in the public section of the class declaration
- The destructor cannot have arguments
- It has no return type
- Destructors cannot be overloaded i.e., there can be only one destructor in a class
- The destructor is executed automatically when the control reaches the end of class scope to destroy the object

```
class Test
{
        public:
        Test( )
        {
        cout<<"\n Constructor executed";
        }
        ~Test( )
        {
        cout<<"\n Destructor executed";
        }
};
void main( )
{
        Test t1,t2,t3;
        return 0;
}
```

Output:

Constructor executed

Constructor executed

Constructor executed

Destructor executed

Destructor executed

Destructor executed